Benchmark of glidein factory monitoring Web presentation (v2_0_beta_4 and v1_6_beta_2)

Igor Sfiligoi
Jan 28th 2009

Introduction

The glidein factory monitoring is composed of four parts; collecting the monitoring information, storing a snapshot in a web accessible XML file, keeping a RRD archive of historical data, and maintaining of a user friendly Web interface.

The maintenance of the Web interface turns to be a quite heavy operation; in order to minimize the security risks, all the data is preprocessed, so that the Web server serves only static web pages and image files. The CPU and IO demands of this monitoring scale with the number of entry points and clients, so it can easily use up all the resources of the glidein factory machine

In order to measure the actual load of such setup, a test setup has been prepared. I have tested the v2_0_beta_4 cvs tag, which uses the same monitoring as v1_6_beta_2. The results are presented below.

Test setup

I performed all tests on cmssrv13.fnal.gov. This is a 4 Xeon 3.2GHz machine with a single IDE hard drive.

The factory was configured with a different number of entry points for different tests.

Different number of frontends were used depending on the test. Each was configured to always ask for 0 glideins; this is the minimum load setup as it tests the load put in the glidein factory by the monitoring alone. For all the tests, a single frontend should be assumed unless specified differently.

Benchmarking results

Out of the box monitoring with 10 entry points

The first test was performed using 10 entry points. A single VO frontend was used.

The load on the machine was very low (see table) and the monitoring performed all the tasks within the minute. In this scenario, the monitoring overhead is completely acceptable.

Load5	~0.5
Idle	>90%
IOWait	<0.5%

Out of the box monitoring with 50 entry points

The next test was performed using 50 entry points. Again, a single VO frontend was used.

The load on the machine was still reasonably low (see table) and the monitoring performed all the tasks within the minute. In this scenario, the monitoring overhead is completely acceptable.

Load5	~3
Idle	~60%
IOWait	<1%

Out of the box monitoring with 100 entry points

The next test was performed using 100 entry points.

The load on the machine was still reasonable (see table), but the creation of Web content was taking up to 20 minutes (see extract from the logs below). Given that the page is supposed to be updated every 5 minutes, this is way too long!

Load5	~15
Idle	~40%
IOWait	~10%

```
[2009-01-27T15:25:13-05:00 7574] Writing lazy stats for qc
[2009-01-27T15:29:20-05:00 7574] Writing lazy stats for logSummary
[2009-01-27T15:32:22-05:00 7574] Writing lazy stats for qc
[2009-01-27T15:47:52-05:00 7574] Writing lazy stats for logSummary
```

Monitoring without locking with 100 entry points

The Web monitoring has internal locks to reduce the load on the system. However, since the previous test showed that it was taking too long for the Web pages to be updated, I manually disabled the locks and repeated the test.

The load on the machine was now much higher (see table), barely bearable. On the other hand, the the creation of Web content was taking approximately 5 minutes (see extract from the logs below), which is still acceptable.

Load5	~90
Idle	~25%
IOWait	~12%

```
[2009-01-27T16:33:42-05:00 19001] Writing lazy stats for logSummary
[2009-01-27T16:33:42-05:00 19001] Writing lazy stats for qc
[2009-01-27T16:36:27-05:00 19001] Writing lazy stats for qc
[2009-01-27T16:37:29-05:00 19001] Writing lazy stats for logSummary
```

Factory without Web monitoring and with 100 entry points

Just me get a baseline, I next completely disabled the production of web content and repeated the test.

The load on the machine was again very low (see table), so most of the load in the previous tests actually came from the creation of user-friendly Web content.

Load5	~0.8
Idle	~80%
IOWait	<2%

Out of the box monitoring with 150 entry points

Restoring the original factory code, I next performed a test using 150 entry points.

The load on the machine was moderately high (see table), but still acceptable.

Load5	~25
Idle	~32%
IOWait	~14%

However, the creation of Web content was taking approximately 1 hour (see extract from the logs below). Given that the page is supposed to be updated every 5 minutes, this is way too long!

```
[2009-01-28T09:44:15-05:00 27672] Writing lazy stats for logSummary
[2009-01-28T09:44:15-05:00 27672] Writing lazy stats for qc
[2009-01-28T10:43:47-05:00 27672] Writing lazy stats for qc
[2009-01-28T10:53:30-05:00 27672] Writing lazy stats for logSummary
```

It should be noted that the rest of the factory was still able to function without problems, though, mostly fitting withing the 1 minute loop. (see extract from logs below)

```
[2009-01-28T10:44:55-05:00 27672] Sleep 60s
[2009-01-28T10:46:02-05:00 27672] Sleep 60s
[2009-01-28T10:47:12-05:00 27672] Sleep 60s
[2009-01-28T10:48:15-05:00 27672] Sleep 60s
[2009-01-28T10:49:17-05:00 27672] Sleep 60s
```

Monitoring without locking with 150 entry points

I again manually disabled the locks and repeated the test.

The load on the machine was now unbearably high (see table), operations like 'cat' taking up to a minute. On the other hand, the creation of Web content was taking approximately 5 minutes (see extract from the logs below), which is still acceptable.

Load5	~85
Idle	~15%
IOWait	~21%

The creation of Web content was a little faster than in the previous test, but still taking approximately 20 minutes (see extract from the logs below). Given that the page is supposed to be updated every 5 minutes, this is still too long.

```
[2009-01-28T07:39:01-05:00 8916] Writing lazy stats for logSummary
[2009-01-28T08:00:37-05:00 8916] Writing lazy stats for logSummary
[2009-01-28T08:11:12-05:00 8916] Writing lazy stats for qc
[2009-01-28T08:21:36-05:00 8916] Writing lazy stats for qc
```

Moreover, the rest of the factory was also getting behind. (see extract from logs below) So this is not a workable solution.

```
[2009-01-28T06:54:19-05:00 8916] Sleep 60s
[2009-01-28T06:59:29-05:00 8916] Sleep 60s
[2009-01-28T07:07:20-05:00 8916] Sleep 60s
[2009-01-28T07:08:32-05:00 8916] Sleep 60s
[2009-01-28T07:39:01-05:00 8916] Sleep 60s
[2009-01-28T08:00:37-05:00 8916] Sleep 60s
[2009-01-28T08:01:39-05:00 8916] Sleep 60s
```

Out of the box monitoring with 10 entry points and 10 clients

I then reverted back to 10 entry points on the glidein factory, but with 10 VO frontend as clients.

In this setup, the load on the machine remained comfortably low (see table) and the monitoring performed all the tasks within the minute. In this scenario, the monitoring overhead is completely acceptable.

Load5	~1.5
Idle	~66%
IOWait	<2%

Out of the box monitoring with 50 entry points and 10 clients

The next test was performed using 50 entry points and 10 clients.

The load on the machine was still reasonable (see table), but the creation of Web content was taking up to 1 hour (see extract from the logs below). Given that the page is supposed to be updated every 5 minutes, this is way too long!

Load5	~18
Idle	~20%
IOWait	~5%

```
[2009-01-28T18:28:26-05:00 29020] Writing lazy stats for logSummary
[2009-01-28T18:37:22-05:00 29020] Writing lazy stats for qc
[2009-01-28T18:59:14-05:00 29020] Writing lazy stats for qc
[2009-01-28T19:30:52-05:00 29020] Writing lazy stats for logSummary
```

The test results were very similar to the 150 entries/1 client test case, so I did not perform any other benchmarks with this setup, assuming I would get a very similar result. I also did not try to scale further, as this was already behind the desired performance.

Conclusions

The benchmarking results clearly show that the monitoring creating user-friendly Web pages does not scale well neither with the number of entry points, nor with the number of clients (VO frontends). With moderate numbers of either, it uses up a significant amount of resources, getting into the way of the normal glidein factory functionality.

This will need to be addressed in the future releases of the glideinWMS, especially in the the v2_X series where multiple clients are expected to be the norm.

References

glideinWMS home page
http://www.uscms.org/SoftwareComputing/Grid/WMS/glideinWMS/

RRDtool home page
http://oss.oetiker.ch/rrdtool/